Recent Developments on Algebraic Specification and Verification in CafeOBJ

FUTATSUGI, Kokichi
二木 厚吉
Japan Advanced Institute of Science and Technology

DCIT 2015, Wuhan, China
16 November 2015
An Important Key Technology in SE
-- A key Technology also to IoT --

Software has became an important infrastructure of our society
Reliability and Security are most important attributes

- Cloud Service Protocols, Electric Commerce Protocols
- International Standards for Important Software Systems
  (standards for on-board computers, etc.)

Verification of Problem Specifications is a Key Technology

Verification of Problem Specs
Problem verification with Proof Scores

Technical Issues

1. Development of specs in appropriate abstraction level
2. Verification methods combining inference and search

CafeOBJ Proof Scores

Innovate Spec Verification and Realize Practical Problem Spec Verifications
Application areas of formal methods (FM)

1. Analysis and verification of developed program codes (post-coding)

2. Analysis and verification of (models/specs of) domains, requirements, and designs before/without coding (pre-coding or without coding)

Successful application of formal methods to the area of (models/specifications of) domains, requirements, designs can bring drastic good effects for systems developments, but it is not well exploited and/or practiced yet.

specification = description of model
Introduction

Speciﬁcation Veriﬁcation and Proof Scores
Comparison of The Three Methods
Cases of Spec and Proof Score Developments in CafeOBJ
Conclusion

An Important Key Technology of Software Engineering and IoT
Application areas of formal methods (FM)
Our Proof Score Approach
Development of proof scores in CafeOBJ

The current situation of FM

- Verification with formal specifications still have a potential to improve the practices in upstream (pre-coding) of systems development processes
- Model checking has brought a big success but still has limitations
  - It is basically “model checking” for program codes
  - Still mainly for post-coding
  - Infinite state to ﬁnite state transformation can be unnatural and diﬃcult
- Established interactive theorem provers (Isabelle/HOL, Coq, PVS, etc.) are not necessary well accepted to software/systems engineers
  - especially in upstream (pre-coding) phase
Our approach

♦ Reasonable blend of user and machine capabilities, intuition and rigor, high-level planning and tedious formal calculation
  • fully automated proofs are not necessary
    good for human beings to perceive logical structures of real problems/systems
  • interactive understanding/description of real problem domains/requirements/designs is necessary

Proof Score Approach
Proof Score Approach

- Domain/requirement/design engineers are expected to construct proof scores together with formal specifications.

- Proof scores are instructions such that when executed (or "played") and everything evaluates as expected, then the desired property is convinced to hold:
  - Proof by construction/development
  - Proof by reduction/computation/rewriting
Development of proof scores in CafeOBJ

♦ Many simple proof scores are written in OBJ language from 1980’s; some of them are not trivial

♦ From around 1997 CafeOBJ researchers at JAIST use proof scores seriously for verifying specifications for various examples
  • From static to dynamic/reactive system
  • From ad hoc to more systematic proof scores
  • Introduction of OTS (Observational Transition System) was a most important step
  • Generate/Check Method is a most recent development
Some achievements of CafeOBJ/OTS proof score approach

| Some classical mutual exclusion algorithms |
| Some real time algorithms                   |
| e.g. Fischer’s mutual exclusion protocol   |
| Railway signaling systems                  |
| Authentication protocol                     |
| e.g. NSLPK, Otway-Rees, STS protocols      |
| Practical sized e-commerce protocol of SET |
| UML semantics (class diagram + OCL-assertions) |
| Formal Fault Tree Analysis                 |
| Secure workflow models, internal control   |
| International standard for automotive software |
| Protocols for Cloud computing              |
A little bit of CafeOBJ history

- KF thought of the basic ideas of CafeOBJ after he participated OBJ project at SRI in 1983-1984, and several design and implementation attempts were done during 1985-1995
- The CafeOBJ development project is fully supported by IPA/MITI of Japanese Government from 1996.4 to 1998.3
  - Six Japanese Companies, Five Japanese Universities, Three Foreign Research Group participate CAFÉ project
  - A book entitled “CafeOBJ Report” was published in 1998 which defines the syntax and semantics of the CafeOBJ language
- Sufficiently reliable and usable CafeOBJ system was available at around the beginning of 1999.
- KAKEN-KIBAN(S) project on CafeOBJ in 2011-2016
- Several groups including KF’s group at JAIST are using CafeOBJ for developing formal methods for various application areas and/or for education of FM
Specification Verification

- Constructing specifications and verifying them in the upstream of system/software development are still one of the most important challenges in system/software development.
- It is because many critical defects are caused at the phases of domains, requirements, and design specifications.
- Proof scores are intended to meet this challenge, and generate & check method is a verification method with proof scores.
- The generate & check method can be applied to any system, but the application to transition systems is most important and interesting.
Verification with Proof Scores

- A system and the system’s properties are specified in an executable algebraic specification language (CafeOBJ in our case).
- Proof scores are described also in the same executable specification language.
- Specifications and proof scores are expressed in equations, and the checks are done by reduction (i.e. rewriting from left to right) with the equations.
- The logical soundness of the checks with correct proof scores is guaranteed by the fact that the reduction are consistent with the equational reasoning with the equations.
System and Property Specifications, and Proof Scores

- For verifying a system, a model of the system is formalized and described as system specifications that are formal specifications of the behavior of the system.
- Functions and predicates that are necessary for expressing the system’s supposed properties are formalized and described as property specifications.
- The supposed property we are considering is either (1) invariant property or (2) (p leads-to q) property.
- Proof scores are developed to verify that the system’s supposed properties are deduced from the specifications.
Verification methods with proof scores includes:
- OTS method,
- CITP method, and
- Generate&Check method.

OTS: Observational Transition System
CITP: Constructor-based Inductive Theorem Prover


In Equations

-- CafeOBJ variables
var S : Sys
vars I J : Pid
-- for want transition
op c-want : Sys Pid -> Bool {memo strat: (0 1 2)}
eq c-want(S,I) = (pc(S,I) = rm) .
--
ceq pc(want(S,I),J)
    = (if I = J then wt else pc(S,J) fi) if c-want(S,I) .
ceq queue(want(S,I)) = put(I,queue(S)) if c-want(S,I) .
ceq want(S,I) = S if not c-want(S,I) .
In Trans/Rewrite Rules

-- wt: want transition
mod! WT {pr(STATE) tr[wt]:
  [Q:Aq r (A:Aid AS1:As) w AS2:As c AS3:As]
=> [(Q | A) r AS1 w (A AS2) c AS3] .}

-- ty: try transition
mod! TY {pr(STATE) tr[ty]:
  [(A:Aid | Q:Aq) r AS1:As w (A:Aid AS2:As) c AS3:As]
=> [(A | Q) r AS1 w AS2 c (A AS3)] .}

-- exc: exit transition with a condition
mod! EXc {pr(STATE) ctr[exc]:
  [(A1:Aid | Q:Aq) r AS1:As w AS2:As c (A2:Aid AS3:As)]
=> [Q r (A2 AS1) w AS2 c AS3]
  if (A1 = A2) .}
OTS Proof Score in 'open...close’s

-- Property of PNAT+: _+_ is associative (+assoc)
-- Proof: By induction on X
-- I Induction Base
select PNAT+.
red (0 + ‘y:Nat) + ‘z:Nat = 0 + (‘y + ‘z).
-- II Induction Step
open PNAT+.
-- induction hypothesis
op x: -> Nat. eq (x + Y:Nat) + Z:Nat = x + (Y + Z).
-- check
red ((s x) + ‘y:Nat) + ‘z:Nat = (s x) + (‘y + ‘z).
close
-- QED
--
Proof Scores in CITP Commands: CITP

-- Property of PNAT+: _+_ is associative (+assoc)
--
:verbose on . -- to see detailed information
-- proof with CITP
select PNAT+ .
:goal{eq[+assoc]: (X:Nat + Y:Nat) + Z:Nat = X + (Y + Z).}
-- by induction on X
:ind on (X:Nat).
-- check
:apply (SI TC RD).
-- QED
--
Proof Scores in Geneate & Check

```plaintext
-- wt: [AQ:Aq r (A1:Aid AS1:As) w AS2:As c AS3:As]
mod WTcoverSet1 {pr(WT) inc(IINVcheck)
op s-wt-0 : -> State .
eq s-wt-0 = [aq r (a1 as1) w as2 c as3] . }
select WTcoverSet1 . red check-iinv(s-wt-0) .
** not successful
mod WTcoverSet2 {inc(WTcoverSet1)
ops s-wt-01 s-wt-02 : -> State .
-- s-wt-01 and s-wt-02 cover s-wt-0
eq s-wt-01 = [empQ r (a1 as1) w as2 c as3] .
eq s-wt-02 = [(a | aq) r (a1 as1) w as2 c as3] . }
select WTcoverSet2 . red check-iinv(s-wt-01) .
    red check-iinv(s-wt-02) .
** successful
```
[OTS] An algebraic/equational programming scheme for writing proof scripts. All proof planning is described and transparent. Flexible and versatile but could be sloppy. Easy to learn and use.

[CITP] Automation of a part of OTS proof scores based on sound and quasi-complete ”specification calculus“. Formal enough and can automate (1) induction and (2) the case splitting induced by conditional equations, but proof planning is not transparent. CITP has a potential to be incorporated into OTS proof scores.

[Gen&Check] Realized as a library in CafeOBJ for automatically generating covering state patterns and checking verification conditions over them. Can be used for covering patterns of any sort. Covering patterns represent split cases and unproved cases are explicit for analyses, which may help to find lemmas. CafeOBJ’s built-in search predicates and the concept of literal constants play important roles. Formal meta-theory for the method is provided.


Properties to be Verified

[OTS] invariant, leads-to, ...
[CITP] invariant, (leads-to, ...)
[Gen&Check] invariant, leads-to

OTS vs. TransRules

[Intuitiveness] Equations < TransRules
Proof Scores

[Flexibility] CITP < Gen&Check < OTS

[Automation] OTS < Gen&Check < CITP

[Formality] OTS < Gen&Check < CITP
Key-Secrecy of PACE with OTS/CafeOBJ, Dominik Klein (dominik.klein@bsi.bund.de) [with OTS] — Key-secrecy of PACE (The ICAO-standardized Password Authenticated Connection Establishment) is proven by first modeling it as an Observational Transition System (OTS) in CafeOBJ, and then proving invariant properties by induction.

A Formal Description of the OSEK/VDX Specification
Hirokazu Yatsu (hirokazu.yatsu@f.ait.kyushu-u.ac.jp) [with OTS+alpha] — A formal description of the OSEK/VDX specification has been developed with an algebraic formalization with right abstraction. Verification of dead lock freeness has been done with the formal description, and the developed formal description is precise enough to verify important properties.

Formalization and Verification of Declarative Cloud Orchestration in CafeOBJ  Hiroyuki Yoshida (yuki.yoshida@jaist.ac.jp) [with Gen&Check+OTS] — The behavior of TOSCA (Topology and Orchestration Specification for Cloud Applications) topologies is specified in CafeOBJ as state transition systems. It is verified that orchestrated operations always successfully complete by proving the transition systems enjoys leads-to (a class of liveness) properties.

To make explicit what good is each proof score method, and establish techniques/methods to coordinate the three methods in an effective way.

To make the message "verification is the best way to get high-quality formal specs" more persuasive. To make it clear what can be obtained out of developments of formal specs and proof-scores.

Make more effective use of versatile CafeOBJ’s module structuring mechanism for giving good structures to specs and proof-scores.

- Generic specs and proof-scores
- Domain specific frameworks for specs and proof-scores
Once a state configuration is properly designed, a large number of patterns that cover all possible cases can be generated and checked automatically. More experiments in the future for identifying a reasonable combination of predicate and pattern/structure oriented case analyses. (i.e. OTS vs. Generate&Check)

Module expressions of CafeOBJ is powerful, and are effective for constructing large specifications/proof-scores with transparent structures. Hoping to provide effective frameworks of proof scores for specific classes of problems.